

## **SCU EVENT PROGRAMMING**

TC-3500 SCU/LDC REV 9202

TC-3550 SCU/LDC REV 9303

TC-550 "BART" CONTROLLER REV 96.12

REV 2.2 03/05/99

### **TABLE OF CONTENTS**

- EVENT TYPES
- SCU EVENT SYNTAX SUMMARY
- OPERATION
- STRING MACRO ENTRY AND EDITING
- ANIMATION MACROS
- ON: EVENTS

This document is intended for the installation and support of Show Control Systems supplied by Triad Productions, Inc. and contains trade secret information regarding proprietary inventions and processes. This information may not be duplicated or revealed in any form without express permission of Triad Productions, Inc.

© 1999 William J. Synhorst/Triad Productions, Inc. All rights reserved.

We believe this information to be correct and accurate at the time of printing. All specifications are subject to change. Triad will not be responsible for any damage related to any use of this equipment or documentation. Please report errors or omissions to Triad Productions.



This document is a complement to the CEVENT Synthesis Event Programming manual. It describes the **Events** subsystem as implemented on the TC-3500/TC-3550 micro-controller Show Control Unit (SCU) or Laser Disc Controller (LDC) systems, in addition to the TC-550 Basic Animation/Real Time "BART" Controller stand-alone systems. When running events on the PC please use the CEVENT portion of this manual.

An EVENT is an action or sequence of actions that occur at a programmed TIME or upon receipt of a trigger or message independent of time which controls show activity. An event may trigger further events, send control messages, begin or complete a sequence of events; all as programmed by the instructions in the events program or "script".

Most of the **event** types are a direct copies to the Synthesis Show Programming System (SPS), although some functions are not available on the smaller playback version (such as lighting cues, multiple output banks, etc.). Conversely, there are many additional commands and modes that are available on the SCU/LDC/BART that are meaningless to the SPS, such as the hardware opto-input and opto-coupled output ports, EPROM cue memory, etc. Whenever possible, compatibility is attempted; however, the difference in power and the environments available create differences in the specific implementations.

The events processing provides for direct access and control of all of the digital and analog outputs. Processing of digital inputs, in addition to control of the serial communications ports, synchronization modes, and all other aspects of the control system.

Logical operations, "X" variables, and a range of timing functions are included to provide as much flexibility as possible. This allows building and customizing the controller for a wide variety of applications, either as a front end to a large complex system, or as a simple, stand-alone, control system.

An ASCII **macro** system is provided to allow message strings or templates to be entered and routed to the serial ports as required in the individual application.

Event **subroutines** may be created for frequent or repetitive operations. Subroutines are also used for program control that occurs asynchronously or independently from absolute show time.

Subroutines may be "armed" for asynchronous or external triggers, or chained together after preset delay times. This feature, combined with animation data CUE macros allows the system to control virtually any number of sub-functions or sub-shows concurrently in frame-locked sync.

Triggers may be activated by remote contact closures, ASCII serial messages, or keyboard function keys assigned to a particular subroutine/sequence.

A typical Event line will generally look like one of the following:

00: 12: 15 Di gOn 005 002

S001. 001 AnVal u 056 254

In the first example, an absolute time is indicated in MM:SS:FF (minutes, seconds, frames) format, followed by the event type, and one or two parameters as required by the type of event. In the first example, the command is to turn on digital channel 5, subchannel 2, at 12 seconds and 15 frames into the "show".

The second example shows the structure of a subroutine; S identifies the line as a subroutine, in this case #1, followed by a ".nnn" which is the sort-order or line number. This particular event says to set analog channel 56 to a level of 254. The event type is either entered as a two digit number, or may be selected using the PgUp/PgDn and arrow keys or a mouse within Synthesis. When an event subroutine is executed, all events with the same subroutine number are executed in order, unless "skipped" by a conditional test.

SCU/LDC/BART events normally are programmed on the Synthesis system and directly uploaded to the battery backed RAM memory of an SCU/LDC or BART system, or burned into EPROM and automatically transferred to the RAM area for TC-3550 based systems. If RAM is used for the events data space, a simple editor is implemented to allow testing and touch up of Event programs.

At the present time, the SCU EVENT capacity is 938 lines or operations, either as timed functions or subroutines in any combination. Some applications may consist of primarily a large number of timed events, where others may be constructed almost entirely as subroutines. The flexibility and power of the Events system can extend and adapt to many implementations and requirements.

Special conditions, such as a reset, external start or stop command, etc. may also invoke pre-defined ON:event routines (explained later in this document).

## EVENT TYPES

The events functions are entered as a decimal number, usually followed by one or two PARAMETERS that fully specify the operation to be performed.

1. **marker** A NOP code that may be used to hold a time or parameter value, but causes no action during execution. Markers may be automatically flagged in Synthesis during execution by pressing the "M" key, and later filled in with the actual operation to be performed at that time.
2. **DigOn** (Digital On) Turns on the digital function specified by the following channel (card) 1-8 and sub-channel to turn on, from 1 to 16/24/32 depending upon the system configuration. (The number of digital channels is set under the C.onfig menu on the LDC/BART main menu.) Note that on BART controllers, the internal digital outputs start on channel 5, subchannel 1.
3. **DigOff** (Digital Off) Turns off a specified channel/sub-channel.
4. **Pulse** Sets up a half second digital pulse (on-wait-off) to the channel (1-8) and subchannel (1-32) specified.

NOTE: A Pulse will only execute properly when one of the SYNC modes (SMPTE, PILOT Internal, etc.) is in effect. Use a DigOn/Wait/DigOff sequence to generate a pulse in ON: Event sequences, or when time code may not be running.

It is possible to use "X" variables for both the card and subchannel numbers for any of the digital event operations. First establish the X array variables for each, then use function 26 (SetXY) to set scalar variables X and Y based on the array index for each variable. To use the value of X as the channel number, use PARM1=255; to use the value of Y for the subchannel, use PARM2=255.

```
S002.001 Xp1=P2 001 005 (sets X(1) = 5)
S002.002 Xp1=P2 002 012 (sets X(2) = 12)
...
S003.001 SetXY 001 002 (sets X=X(1), Y=X(2))
S003.002 Di gOn 255 255 (turns on channel 5, subchannel 12)
```

5. **AnValu** (Analog Value) - Forces the indicated channel (PARM1 = 1-64) to the value specified in the second parameter, from 0-254. Specifying the PARM1 analog channel number as 255 allows the most recently set "X" variable to be used as the channel number. Using a value of 255 for PARM2 allows the most recently set "Y" variable to be used as the LEVEL.

Examples of using AnValu:

```
01:15.12 AnVal u 001 000 (sets channel 1 to value of 0 at 1:15.12)
S003.001 AnVal u 015 200 (sets analog channel 15 to a value of 200
when subroutine 3 is executed)
S004.001 Xp1=p2 003 016 (sets X(3) = 16)
S004.002 Xp1=p2 004 128 (sets X(4) = 128)
...
S005.001 SetXY 003 004 (sets X=X(3), Y=X(4))
S005.002 AnVal u 225 255 (sets analog channel 16 to a value of 128)
```

Note that using AnValu will cause a "bump" or immediate change to the output value. It is possible to generate a smooth fade or ramp using FadeRt [34] and FadeTo [35] events described below.

**NOTE:** The TC-3500/3550 SCU may only address 64 analog channels, while the current limit for a BART Controller is 16 channels.

**6. System** Allows special system functions to be executed in an event. The first parameter indicates the system function type:

VALUE	FUNCTION
1	RESET
2	Force update of all digital/analog outputs
3	Set SMPTE sync mode
4	Set INTERNAL (Crystal lock) time base
5	Set PILOT (VIA 1/PA1 60 Hz) time base
6	Set Wait/internal (Cycle mode)
7	Set Wait (EOT) OLD!!
8	Generate SMPTE on startup (BART only)
9	Force ENDEV (4) to occur
11	STOP pulse
12	PLAY pulse
13	REWIND pulse
14	FAST pulse
15	ON Opto 1
16	
17	
18	
19	OFF Opto 1
20	
21	
22	
30	Set Sony handshake mode [P2=128] or clear [P2=0] [for [10] COM1: event only]
34	Purge ALL arm delays, frame delays, and triggers
35	Purge trigger buffers (ASCII or hardwired triggers)
36	Scan memory and building animation macro ID table. PARM2 must be set to the starting bank or EPROM. Only matching animation macro Ids will be overlaid. THIS MUST BE DONE BEFORE a MacOn (event 36) is used!
37	Kills/terminates all running animation macros

**NOTE:** System commands 11-22 controlling the "deck" opto outputs are unique to the TC-3500 and TC-550 BART controllers, and do not apply to the TC-3550 or Synthesis programming systems.

For TC-550/BART systems, SMPTE may be decoded directly by the hardware, by feeding 30 FPS, non-drop longitudinal time code into the SYNC connector, and placing the SYNC jumper to the SMPTE strapping. 60 HZ. pilot, video sync, or composite video may be used for all systems in PILOT mode.

**7. Pause** Beginning with release 96.12, PAUSE now acts as a delay within a subroutine between "sorting" or "point" numbers with PARM1 being the number of SECONDS to PAUSE and PARM2 being the number of FRAMES. This allows for complex delay structures to be built **within** a subroutine (no need for "bucket brigade" delay subroutines). This should be used with caution because there is currently no way (on the LDC/BART) to kill a (looping) subroutine with embedded pauses.

S001.001 Pulse 005 001  
 S001.001 Pause 010 015  
 S001.002 Pulse 005 002

In this example, Digital output 5:1 would pulse and 10.5 seconds later 5:2 would pulse.

- 8. **E.O.T.** Sets an internal flag indicating that the end-of-tape is approaching; about 1/2 second after data or SMPTE time code disappears, a special ON EOT event will be processed. This would normally be placed after an END enable, and will only be reached by the (last physical) show which has enough extra time code for this event to occur. One possible use is to issue a tape home or recue command.
- 9. **EndEnb** As described above, sets a flag indicating that the end of the show is expected, and what sequence to perform once the show is over (i.e. stop deck). Both EOT and END enables are also used during data-on-tape encoding, and are equivalent to S8 and S9 respectively.
- 10. **Com1:** Allows an ASCII character or ASCII string macro to be sent to the auxiliary COMM port on the TC-3510 I/O controller, or the "C" port of a TC-550 BART controller. If the second parameter (PARM2) is "0", the PARM1 corresponding ASCII character will be sent as a command code to the COM1: port. If the second parameter is not 0, the PARM2 parameter is used as a pointer to the macro string to send (see below).

- There is now a soft switch (system command 30) that allows full handshaking with Sony LDP series laser disc players through the COM1: port.

Examples:

As a timed event:

01: 05. 01 COM1: 049 000 Sends ASCII "1" when time code=1:05.01

In a subroutine:

S001. 004 COM1: 000 012 Sends ASCII macro string #12

S001. 005 COM1: 013 000 Sends a CR (ASCII 13) character

S002. 006 COM1: 000 000 Sends a "Null" character

- For a BART controller, COM1: is labeled as SER C (DB-9F style connector) on the back panel. Thus, for BART applications, think of event 10 as SER C.

- 11. **Com2:** Sends the macro string (PARM2) to the COM2: or terminal port, in a similar manner as described above. A special case exists IF the first parameter/character is \$FE (254). This is the lead in character for a trigger/message to a host (Synthesis) control system. The \$FE (254) character is sent. Then, if the second parameter is not 0, the ASCII character represented by the second parameter is also sent as a literal ASCII byte. If PARM1 is 0, then the ASCII macro string specified in PARM2 is parsed and passed to the serial port.

Example:

S002. 002 COM2: 000 012 Sends ASCII macro string #12

S002. 003 Com2: 013 000 Sends a carriage return (ASCII 13)

S003. 001 Com2: 254 000 Sends binary character 254 (\$FE) only

S004. 003 Com2: 254 065 Sends 254 (\$FE) followed by ASCII "A". This would be used to send a trigger/message to a host system (i.e. Synthesis or MBPS system), using a single event rather than two consecutive, single byte commands.

- **For a BART Controller, COM2: is labeled as SER A (modular style connector) on the back panel. Thus, for BART applications, think of event 11 as "SER A." This is the port normally used for all terminal, diagnostic, and uploading operations, and to also send time code and trigger messages to a host (Synthesis) computer system if used.**

- 12. QCom1:** Works exactly the same as function 10 (COM1:), except that all comm requests are spooled into an internal buffer and released during interrupt processing. This allows long messages or communications at slower baud rates without holding up the system during the message. Characters are queued at approximately 60 CPS.
- 13. Com3:** Same as COM1: and COM2: commands, except that the serial port is on a second TC-3510 I/O card. For TC-550 BART Controllers, this is the port labeled "SER B" on the back panel (modular style connector).
- 14. Com4:** MIDI - For TC-3500 systems, this command talks to a modified TC-3505 to send MIDI command strings. For TC-550 BART Controllers, this command is used to send ASCII strings or messages to the serial port labeled "SER D".
- 15. XCom:** Sends the macro string defined in PARM2 to the TC-3518 serial ports bitmapped in PARM1. Bit 0 corresponds to port 1, through bit 7 to physical port 8. This applies to TC-3550 and BART-based systems that are equipped with one or more TC-3518 serial controller interfaces.
- 16. Disply** Displays the text of the specified macro number in parameter 2 to the COMM (terminal) port. All characters of the specified macro will be sent to the terminal (COMM) port. This feature is provided for compatibility with the Synthesis system, and is generally not required.

- ***On TC-550/BART systems, a display causes the first 16 characters of an ASCII macro to be displayed on the second line of the LCD display.***

- 17. APatch** Allows dynamic reassignment of the physical **ANALOG** output channel number (an actual D/A converter 1-64) from the logical channel number used during programming. Normally, the logical and physical channels are assumed to be assigned one-for-one, and no repatching is required. To install a patch, use this event #, followed by the PHYSICAL OUTPUT channel (TO) and then the LOGICAL channel number (FROM).

```
00: 12. 13  APatch 003 001 } devices connected to outputs
00: 12. 14  APatch 004 001 } 3 and 4 will follow channel 1
```

Note that multiple physical **output** channels may follow a single programmed or "logical" channel, thus simplifying programming and reducing the number of cues needed. It is further possible to reassign patches ("soft patch") at any point or multiple times in a show. Currently, X and Y variable assignments are not available for creating soft patches.

- 18. SETPAG** PARM2 sets the base page of eight (8) consecutive pages of 64: 32 byte macro strings FOR THE NEXT MACRO used. After an ASCII macro is sent, the page returns to the default address (\$38, or 54 decimal). This allows for MANY, MANY groups of 64: 32 byte or 128: 16 byte strings, addressed as before in any of the comm or display commands.

- 19. Parall** Only available in CPU Events.

## X VARIABLES

Beginning with release 6.8, up to 64 "X" variables are permitted. In most cases, the X is selected by PARM1 (from 0-63); for the conditional branches, X is the last value modified or SET.

**CAUTION:** When a subroutine, armed event, or delay-triggered subroutine occur, the current X value is saved prior to the call, and restored afterward. This is to preserve the X value should an asynchronous event happen between setting and testing "X". If in doubt, put a "SET X" (26) event code just prior to one of the test/branch events.

Error messages are given if an undefined subroutine is called, or if you attempt to call a subroutine from within a subroutine. In the future, nested routines may be allowed.

**20. Xp1=p2** X(PARM1)=PARM2 Sets variable array element X[PARM1] to the value (0-255) of the second parameter. The scaler "X" variable used for conditional testing, modification, or variable event parameters is also set.

- **By setting the high bit of the first parameter to '1' (128+ X element), the second parameter may be one of the VARIABLES defined in the following table.**

P2	VARIABLE		
0-63	X() Array variable (allows a variable to be copied)		
65-128	Analog value of channel in PARM2 (P2-64)		
128			
129	XX		(SCALER X)
130	XY		(SCALER Y)
131	HOURS	0-23	(TIME CODE, WITH CURRENT OFFSET)
132	MINUTES	0-59	
133	SECONDS	0-59	
134	FRAMES	0-29	
135	DAY		(REAL TIME FROM SYSTEM CLOCK)
136	HOUR		
137	MINUTES		
138	SECONDS		

Note: The above values will only be valid if a real-time clock module is installed. Otherwise, this is the relative, 24 hour time since the system was restarted.

P2	VARIABLE	
139	POINTER SET BY [40] SETPTR into macro or PA-422 buffer	
140	RANDOM Random # 0-255 (can be clipped)	
141	INPUTS 1-8	The current status of inputs 1-8
142	INPUTS 9-16	As above, but inputs 9-16
143	INPUTS 17-24	This allows a parallel "byte"
144	INPUTS 25-43	of data to be captured/processed
145	INPUTS 33-40	in "X" variables, without using the
146	INPUTS 41-48	edge trigger processing of discrete bits.

Caution: The values 192-255 for P1 are not currently defined, but have been reserved for future expansion of X variables. DO NOT use these values for any X variable!!

**21. X=X+p2** Adds the VALUE of P2 to the current "X" variable. If P2=0 or 1, the effect is to increment X by 1 (for compatibility). If P2>1, then the value of P2 is added to X. The scaler "X" variable is set to the new value, so that a test/branch may be correctly executed directly after an "X=X+1" or "X=X-1" operation.

- 22. **X=X-p2**      **Subtracts the VALUE of P2 from the current "X" variable. If P2=0 or 1, the effect is to decrement X by 1 (for compatibility).**
  
- 23. **ModifyX**      Performs a logical or mathematical operation on the current X variable (as set with event 26, SetXY or event type 20 X(P1)=P2).

Modify the current 'X' variable based on the operation code in the first parameter, using the data (if appropriate) in parameter 2. The operations are listed in the following table:

P1 = Modify algorithm, P2 = DATA (if required)

PARAM1	FUNCTION	DESCRIPTION
*0	(FUTURE)	
1	OR	Logically ORS value with
2	AND	Logically ANDS value with
3	XOR	Perform an EXCLUSIVE OR with
*4	SHL	Logical shift left 1 bit **
*5	SHR	Logical shift right 1 bit **
*6	ROL	Rotate left 1 bit **
*7	ROR	Rotate right 1 bit **
*8		
*9		
A	MIN	Forces a minimum value of (clipping)
B	MAX	Sets a maximum value of
*C	LOG	(Fixed translate to log table for audio)
*D	XLT	Translate from 256 byte table 2
*E	USR	Translate from table 3
F	RND	Sets X to a RANDOM number

**\* Not implemented at this time    \*\* may be changed to P2 # shifts**

Note that a X=P2 (event type 20) using VARIABLE PARM2 of 129 can be used to resave an X variable after it has been modified.

- 24. **If X =**      **If X = (PARAM1) ELSE SKIP (PARAM2) EVENTS**  
This is a conditional test that may be used to process a series of events based on a counter or toggle. "X" is the value of the last X modified or SET (26).

- 25. **If X #**      **If X # (PARAM1) ELSE SKIP (PARAM2) EVENTS**  
Same as above, except the following events are executed if X<>PARAM1, otherwise (PARAM2) events are skipped.

*Beginning with releases later than 03/88, a new method of conditional events execution is implemented.*

- 26. **Set XY**      Forces the current X variable to X(PARAM1) and Y to X(PARAM2) without changing the value of either X(PARAM). Up to 64 "X" variables are allowed, from 0 to 63. The X and Y values may then be used as variable parameters to many other event types (such as digital or analog channels, timing values, etc.), or modified and tested in a number of ways.

- |27. **if CYCLE**      Tests the cycle mode switch, skips PARAM2 events if the cycle mode is not in "hold" status.

- 27. **if RUN**      Only available in CPU Events.

- 28. **if HR=**      Only available in CPU Events.

- 29. if HR#** Only available in CPU Events.
- 30. Do\_Sub** Performs an immediate call to an event Subroutine. Subroutines are built from the top of the events list and work down. They are entered in the format of Snnn.sss, where nnn is the subroutine number (1-250), and sss is a sequence number (used for jumping into the middle of a subroutine using Do\_Sub, separating delayed sections of a subroutine using the PAUSE command and for clarity and sorting purposes). Sequential events are executed as long as the subroutine number (nnn) equals the Do\_Sub number, thus there is no explicit "return" required. At this time, Do\_Subs may be nested (i.e. called from within subroutines).
- Unlike ON:events, subroutines do not output data for each event executed. Once all events are scanned, the analog and digital channels are updated once for each frame. If absolutely required, a System (6) function can be used to force an update to all digital and analog output channels. Normally, ALL subroutines and events will be scanned, processed, and updated for each frame, and all outputs will be updated at the nominal frame rate.
- PARM2 is now (96.12 and later) used to jump into a subroutine at a position other than the beginning and uses the "sorting" number (ie: S001.XXX) for the position to jump to within the subroutine.
- ***Normally, PARM2 MUST be 0 to execute a subroutine. If PARM2 is GREATER than PARM1, then a RANDOM subroutine between PARM1 and PARM2 inclusive will be performed. Although this still works, we prefer that you use the random number generator and the MIN and MAX functions to perform random subroutine calls.***
- 31. ArmSub** The subroutine (PARM1) will be executed if the condition (PARM2) occurs while the system is executing.
- Triggers (PARM2) 1-64 represent the hardware rising edge or (active high) inputs from TC-326 Input Modules or the BART input section, while triggers 65-128 represent the falling (trailing edge) of external inputs.
- "Software" triggers may be input as ASCII sequences through the RS-232 COM (terminal) port, e.g. "001T" would queue up trigger #1.
- Up to 64 subroutines may be armed at a time, and all armed conditions are cleared with a R.eset (hardware or software).
- Currently, no error message is reported for an attempt to perform an unarmed trigger, or for a trigger armed to a non-existing subroutine.
- Only one subroutine may be armed to any given trigger at one time. To disarm a trigger, either specify a subroutine (PARM1) with a 0 trigger, or a 0 subroutine for the trigger to disarm (PARM2).
- 32. ArmSec** Sets up a subroutine number (PARM1) that will execute after (PARM2) seconds of show code. Note that the delay is based on the actual show time code, and NOT the real time clock. It is legal to arm a delayed-subroutine from within a subroutine. To cancel an pending ArmSec delay, specify the subroutine number (PARM1) with a delay time of 0 seconds. The wildcard for PARM2 is 255 for BART/LDC systems.
- 33. ArmFrm** Similar to ArmSec, except that PARM2 specifies the number of FRAMES to delay (instead of seconds). As with ArmSec, an ArmFrm may be canceled prior to execution by specifying 0 as the (frame time) parameter, and subroutines may be daisy-chained or cascaded with one or more ArmFrms.

It is possible to use the X and Y variables from a SetXY [26] as parameters for ArmSec and ArmFrm. Use 255 for PARM1=X or for PARM2=Y to specify a variable for a subroutine or timing parameter. This allows generic patterns or templates for programs to be created, passing timing parameters as variables.

**34. FadeRt** Sets up an auto fade on an analog channel (PARM1). PARM2 is the rate at which the fade will occur.

- ***The first parameter is now a "dummy" argument. Once a fade rate has been set, multiple analog channels may be assigned to FadeTo new levels at that rate. Note that the rate is inversely related to time, i.e. a lower rate is a longer time. A rate of 32 corresponds to \_\_\_ ms.***

**35. FadeTo** Sets a target level (0-254) in PARM2 for the analog channel specified by PARM1 to fade to. The rate that will be used to get from the initial to target levels is specified by the last performed FadeRt (fade rate) command. If PARM1 is specified as 255, then the current value of "X" is used as the channel number, and if PARM2 is set to 255, the current value of "Y" is used as the target level. Note that the current value of X and Y are set using event type 26 (SetXY).

**36. MacOn** Starts (or restarts) the animation macro cue file specified in PARM1 (PARM1 = 1-32, mod 32) for execution concurrent with any other animation (and/or events) cues.

Animation macros may either terminate themselves after completion (\$FB 00 "Kill") in the cue file), loop for chasing or "random" applications (\$FB nn), or be forced to terminate with event type 37 (MacOff). If P1 is specified as 255 (\$FF), then the current value of "X" is used as the animation macro number to start. Note that animation macros may be started as the result of a timed event (to play in sync with time code), or from an event subroutine, as the result of an asynchronous trigger. The animation file is played sequentially in frame sync to the current sync source (SMPTE, pilot, or internal), but it will not "chase" a change in time code.

NOTE: A SYSTEM (6) 036 <rrr> event must be executed at some point prior to attempting to issue a MacOn. This is required to scan the cue ROM data in order to locate the start points and ID's for all of the macros. When using a TC-3508 memory card, PARM2 specifies the EPROM (rrr) to begin scanning. For BART systems, use 1 to specify cue EPROM #1.

In the BART, the SYSTEM (6) 036 001 must occur as an event not as part of the on reset.

**37. MacOff** Terminates the macro (PARM1) started from a MacOn event. If parm 1 = 255 (\$FF) then the current X variable is used as the macro number to stop. Note that the analog and digital channel status will freeze at their present values, and that any channels that need to return to a home or preset position should be forced with a series of Events or a "home" animation macro routine. A RESET operation disables all macro routines! Likewise, all running macros may be terminated with a SYSTEM (6) event with PARM1 = 37 (see SYSTEM events).

**38. JamClk** Forces the system to jump to the minutes and seconds specified by PARM1 and PARM2. This is useful only when running in INTERNAL or PILOT sync modes, as SMPTE time code will force the system to update to the time code point. If the jam time is greater than the current show clock, all timed events will normally be executed (immediately) up to the new time. If the jam time is earlier than the current time, the effect is to execute all events starting at 00:00.00 up to the new time.

**39. SetLok** Used as the FIRST event in an event SUBROUTINE. The first parameter is a lock CHANNEL, from 0 to 15. If no other locks are set on this channel, the subroutine is allowed to execute, and the lock associated with the channel number is SET. If the

channel is LOCKED, the subroutine is aborted, and queued for execution when the channel becomes UNLOCKED. This allows resources (like the COMM port) to be used exclusively until the operation has completed.

- 40. **ClrLok** (Channel)=PARM1 to unlock. Note that this command decrements the lock-use counter for a particular channel, and may not necessarily clear it.
- 41. **SetPtr** PARM1 is the low byte, PARM2 the high byte of an address in memory. This sets up a pointer for subsequent peek or poke operations.
- 42. **Poke** ***PARM1 is a data byte to poke (force) at the address set by a SetPtr, UNLESS PARM2 is non-zero. If PARM2 is > 0, then IT is used as the value of the X variable to poke.***
- 43. **Peek** PARM1 is the X variable (1-63) which will receive the data byte read from the address set by SetPtr (41). PARM2 may be used as an offset from the base address, which allows different registers or offsets in a buffer to be accessed. If PARM2 is specified as 255, the scaler X variable is used as the offset. Note that 'X' is replaced with the value returned by the peek operation!
- 44. **LDPcue** Only available in CPU Events.
- 45. **LtCue** (Reserved for lighting cues in Synthesis)
- 46. **--SB--** This is used in Synthesis to define the start of and name for a subroutine. It is optional, and effectively acts as a no-operation when encountered in an events script in an LDC/SCU/Bart system.
- 47. **Break\*** Sends a BREAK to the serial port specified by PARM1, for a duration (temporarily!) specified by PARM2.

PARM1	FUNCTION
1	TERMINAL (BART "SER A")
2	DATA (BART "SER B")
3	COM3
4	COM4
11-18	TC-3518 ports 1 to 8
21-28	TC-3518 ports 9 to 16 *

Note that this is a system specific event type, and may have other functionality in different systems.

- 48. **DigByt** Allows a bitmapped output of PARM2 to eight sequential digital channels, starting with the channel in PARM1. The channels align on 8 bit boundaries, beginning with channel 1, subchannel 1. This command may be used for parallel data output, and for matrix selection outputs such as a laser disc controller using a TC-388 Multiplexor card. The following values are possible for P2:

PARM2	FUNCTION
0	Use the current 'Y' variable as specified in the last SETXY
1-64	Value for X variable (PARM2)
255	Uses the current value of the 'X' variable

03/22/93 The order has been reversed so that the LSB of the value is output on the lowest channel number.

Note that the behavior of DigByt is quite a bit different when running on an LDC or BART than when running under Synthesis on the PC. Here, the second parameter is always an "X" variable, not an absolute value!

- 49. CDRom**      Used only in CPU Events.
- 50. Sound**      Used only in CPU Events.
- 51. SndTrg**      Used only in OS9000 system.
- 52. Send X**      Used only in OS9000 system.

## SCU EVENT SYNTAX SUMMARY

	EVENT	PARAM1	PARAM2	COMMENTS
0	-----			Unused event
1	marker			Reserves a location in the events file
2	DigOn	(card)*	(chan)**	Turns on specified digital channel
3	DigOff	(card)*	(chan)**	Turns off specified digital channel
4	Pulse	(card)*	(chan)**	.5 second pulse to digital channel
5	AnValu	(chan)*	(value)**	Force analog channel to preset value
		* X	** Y	Can be replaced with "variables"
6	System	(function)		Operating system special functions
7	Pause	(delay)		Hangs system for delay or CYCLE switch
8	E.O.T.			Sets END OF TAPE enable flag
9	EndEnb			Sets END OF SHOW enable flag
10	Com1:	(\$ or 0)	(macro)**	Sends macro in PARM2 or ASCII in PARM1 to COM1:
11	Com2:	(\$ or 0)	(macro)**	Sends macro in PARM2 or ASCII in PARM1 to COM2:
12	QCom1:	(\$ or 0)	(macro)**	Same as function 10, but queued
13	Com3:	(\$ or 0)	(macro)**	Same as COMM commands, but sent to COM3:
14	Com4:		(macro)**	Sends out MIDI to LDC's or strings to BART "D"
15	XCom:	(port map)	(macro)**	Send macro to port (PARAM1) (LDC/3518 only)
16	Disply	(line)	(macro)**	Display text of macro to screen (COM1:)
17	APatch	(output)	(logical)	Assign OUTPUT channel from LOGICAL channel
20	Xp1=p2	(index)	(value)	Set variable/counter to an absolute value
20	Xp1=p2	(128+index)	(variable)	Set X(P1-128) to a special value
21	X=X+p2	(index)	(value)	Bump X if (value) = 0 or 1, else add PARM2
22	X=X-p2	(index)	(value)	Reduce X if (value) = 0 or 1, else sub PARM2
23	ModifyX	(func)	(parm)	Modify the current X variable based on (func)
24	If X =	(value)	(skip)	Skip steps unless X = constant
25	If X #	(value)	(skip)	Skip steps if X <> constant
26	Set XY	(parm1)	(parm2)	Sets X=X[PARAM1], Y=X[PARAM2]
27	if CYCLE			
30	Do_Sub	(routine)		Calls subroutine at time specified
31	ArmSub	(routine)	(trigger)	As above when async event trigger occurs
32	ArmSec	(routine)	(seconds)	Activates event sub after seconds delay
33	ArmFrm	(routine)	(frames)	Same as ArmSec, PARM2 = frames to wait
34	FadeRt	(channel)	(rate)	Setup fade rate for a FadeTo on analog channel
35	FadeTo	(channel)	(target)	Initiates fade to target level at FadeRt
36	MacOn	(1-32)		Begins an animation cue subroutine/macro
37	MacOff	(1-32)		Forces a kill of a macro (1-16)
38	JamClk	(minutes)	(seconds)	Jams the internal clock to new time
39	SetLok	(channel)		At START of subroutine to lock a resource
40	ClrLok	(channel)		Allows a queued, locked routine to run
41	SetPtr	(low)	(high)	Sets the address for peek/poke operations
42	Poke	(data)		Pokes the data byte (P1) into last SetPtr
43	Peek	(X var)	(offset)	Sets X variable (P1) from address set by 41
46	--SB--			Defines start of and name for a subroutine
47	Break*	(comm)	(length)	Sends a break line status to the comm port
48	DigByt	(channel)	(X/Y var)	Bitmapped output of X/Y starting at CHANNEL

Events preceded by a "#" will NOT execute in "catch up" mode after a search (NOT IMPLEMENTED ON TC-3500 AT THIS TIME). Events preceded by "?" will or will not execute based on the result of the last conditional test of "X", CYCLE, or HRS.

## OPERATION

Normally, all timed events are executed in order, up to the current time code location. In many cases, and especially during programming, it is not necessary to execute certain cues (i.e. strobe flashes) unless the show is being run end to end. Placing a "#" (pound sign) key in front of the event type number will set a flag which causes the event to be skipped during searches, and executed only if the time code matches the event time. Internally this is represented by bit 6 (decimal weight 64) added to the event type.

SUBROUTINES are executed in the following cases:

- A "Do\_Sub" (event type 30) from within a timed file.
- A trigger or input is received that matches that of an ArmEvt, in which case the subroutine associated with that trigger is performed. A trigger can be either edge (rising or falling) of a physical input, i.e. triggers 1-64 occur when an input becomes active, and triggers 65-128 occur when an input makes an "off" transition. A trigger may also be received as a serial message from the terminal COM port, in the form of "nnnT", where "nnn" is the ASCII string corresponding to the trigger message, followed by an ASCII "T", which performs the actual trigger. Note that any armed event may be executed in this fashion, including those that may be intended for hardware triggers. Thus, it is suggested that "software" triggers be assigned to the range of 129 through 255 to avoid conflicts with possible hardware inputs.
- The number of seconds (ArmSec) or frames (ArmFrm) have elapsed since being armed as a time delay.
- A schedule entry matches the day and time as read from the real time clock. At the designated day and time, the subroutine will be performed.

No error will be reported if a called subroutine does not exist, basically, nothing will happen! (But the system will not crash, either!)

## STRING MACRO ENTRY AND EDITING

Up to 56, 32-byte character strings may be entered as Macros to be sent to any of the available COM ports, or to the front LCD display on a TC-550 BART Controller.

(The space previously used for the last 8 of 64 macros is now used to define animation cue subroutine offsets, analog channel configuration, banks, and other system parameters.)

In order to shorten the string (less than 32 characters) and to allow comments in the macro field, the "\" (backslash) character has been reserved as a delimiter, which ends transmission of text from the "\" to the end of the line.

As the "ESC.ape" key would normally be used to exit the editor, it is possible to transmit an ASCII escape sequence (27d, 1Bh) by substituting the "|" (shift-backslash) character.

Other special ASCII codes may have to be sent using the direct form of the event (that is, where PARM2 is not 0), if they cannot be entered from the keyboard. An example is a CR character.

It is possible to send any ASCII character (00-FFh) from within a macro string by using the form "/nnn" where nnn is the decimal equivalent of the character to send. The next, non-numeric character is used as a break character AND IS SENT, with the exception of the letter "X". In this case, the number just parsed will be used as the X variable to use as the ASCII character, allowing variable computed characters to be sent.

Note that the X variable will be picked up out of the array (00-63) of X's.

- 10/27/89 - Similar to above, any ANALOG channel value may be used to form a character. The syntax is "/nnA", and as above, the "A" will not be sent.
- 10/27/89 - The macro string routine now calculates a running checksum on all of the characters actually sent (including those of analog or X variables), and may be sent as a character by using "/C" in the macro.
- 12/16/91 - Added INVERTED checksum (i.e. MOD 256) by using a /c (lower case "c") to accommodate Digicart and other bizarre formats.
- 12/16/91 - Added % token to ASCII Macro interpreter - This character will CLEAR (reset to 0) the current running checksum, (and will not be sent!).

If RAM is present in the TC-3500, it is possible to view and/or edit the MACRO sequences. If an EPROM is used, it is possible to view the current definitions, but no changes may be made.

Press P (Program) from the main menu  
 Press E (Event editor) from the program menu  
 Press M (Macro string editor) from the event menu

The first macro will be displayed, followed by "<01", indicating the current macro number.

Use the "<," or ">." keys on the terminal (unshifted) to scroll forward or backward to the desired macro. (There currently is no rollover test for above 56 or below 00.) To change the current definition, press either the [SPACE BAR] or "E" keys. Type in the new macro definition, and terminate with a [RETURN] if you wish the rest of the line to be filled with spaces, or [TAB] if you wish to keep the remainder of the line intact. The current macro will be redisplayed. Press the [ESC] key to return to the event menu.

Note: In an uninitialized RAM, it is possible (and quite likely) that random characters will be present in the macro definitions, such that during editing, the characters will cause the terminal/emulation used to talk to the SCU to perform unpredictable sequences. Either use the monitor to fill in "safe" characters, or U.pload null macro data from the Show Programming System. When preparing an EPROM, the best fill character is \$FF, such that new data could be programmed in the existing EPROM.

- **Beginning with release 88.06, all ASCII based events now use PARM2 to determine whether to send a string macro (PARM2 > 0), or PARM1 as a direct ASCII character. This allows a NULL character to be sent.**

Beginning with release 89.02, the number of macros strings may be doubled by referencing the macro as #129 to 240 (see appendix C). A 32 byte string can be broken into two strings of up to 16 bytes each. Odd numbers reference the first 16 bytes, even numbers the second 16. All character translations are still in effect, and the string is still terminated with "\". Macros 1-64 still will reference the normal 32 byte string.

Beginning with release 89.02 it is now possible to send a string macro based on one of the X array variables by specifying 255 for PARM2. Be sure to explicitly establish X (using SetXY) to set the static variables from the array.

## **ANIMATION MACROS**

Animation data is normally loaded and reproduced from a TC-3505 RAM/ROM Expansion Interface Card for card-frame based systems. In a TC-550 "BART" Controller, animation data is stored in EPROM or FLASH-ROM in on-board sockets. Animation data (real-time programming, plus converted lighting cues, etc.) is stored in a highly compressed format, and is designed for minimal memory utilization, while providing total reproduction of the analog, digital, and special event operations required for show playback and performance.

## ON: EVENTS

A special set of functions has been defined for the following special **events**, those at which the actual time is indeterminate or when time code may not be present:

- ON RESET** Defines what special actions are to occur when the TC-3500 is restarted, or the R.un (reset) command is given from the computer keyboard.
- Normally when the R.eset option is used, the clock is jammed to 00:00.00, the cue file and event cues are placed at their initial positions, **and all channels are reset to off.**
- ON START** Defines the functions to execute if a remote operator START button is pressed, or the "S.tart" option is selected from the computer keyboard. Normally programmed to invoke a sequence to start a show, including a "S" (start deck) command to the TC-500 SMPTE Time Code Reader/Controller.
- ON ABORT** Performed if an external switch closure or "A.bort" is pressed. Can be programmed for an orderly shut down in case of a break down. Normally, this could be programmed to recue the system and allow a restart.
- ON END** Requests the actions to be taken if an "END ENABLE" event has been entered into the EVENT list **and** the SMPTE code is lost for more than one second (at the end of the show). Used to define the sequence at the end of a show, but not necessarily the physical end of the reel, cassette, or loop.
- ON EOT** Analogous to "ON END", except this sequence marks the physical end of the tape (or last show on a reel), and can be used to issue a rewind or recue to the beginning of the film or tape. One way to differentiate the last show when running under SMPTE is to include enough extra code in the last show to cause the ON:EOT event to be processed, overriding an END.

ON END and ON EOT are incorporated as a fail safe against momentary (deliberate or accidental) loss of time code, stopping the tape, or whatever. Only when an event has enabled and expects the actual end of the show and/or reel of tape is the appropriate event sequence invoked.

These special **events** are located above the normal stack of 1000 time based event cues, and are referenced by, and at the time the specific condition occurs.

*NOTE: The **ON:ERROR**, **ON:PAUSE**, and **ON:INIT** sequences are not implemented in the TC-3500 software.*

**NOTES**